Worksheet 3 – Random Forests

Worksheet 3

Camille Mondon

February 26, 2024

The aim of this worksheet[1] is to learn how to use random forests with R. This requires the package `randomForest`, that should be installed and loaded. The method is illustrated on examples taken from the UCI Machine Learning repository. The corresponding datasets are available in the R package `mlbench`:

```
library(randomForest)
library(mlbench)
```

Random forest: training, interpretation and tuning

Exercise 1. In this exercise, we use random forests to discriminate between sonar signals bounced off a metal cylinder and sonar signals bounced off a roughly cylindrical rock. The data are contained in the dataset `Sonar`:

```
data(Sonar)
```

1. Using

```
?Sonar
```

tell how many possible predictors the dataset includes. What are their types? How many observations are there?

2. Randomly split the data into a training set (with size 100) and a test set (with size 108).

3. Use the function `randomForest` to train a random forest on the training set with default parameters (use the `formula` syntax described in `?randomForest` and the options `keep.inbag=TRUE` and `importance=TRUE`). What are the OOB predictions (for this training set)? Deduce the resulting OOB error. Display the confusion matrix associated with OOB predictions.

4. Using `?randomForest`, describe what is in `ntree`, `mtry`, `votes`, `oob.times`, `err.rate`, and `inbag` in the random forest object from the previous question.

5. What is the OOB prediction for the 50th observation in the training set? What is the true class for this observation? How many times has it been included to design a tree? In which trees has it been included? How many (OOB) trees voted for the prediction 'R' for this observation?

6. Using the `plot.randomForest` function, make a plot that displays the evolution of the OOB error rate when the number of trees in the forest increases. What do the various curves represent? Add a legend and comment.

[1] The content of this worksheet is strongly based on a worksheet designed by Nathalie Vialaneix and Davy Paindaveine.

7. Using `?randomForest` again, describe what is in `importance` in the random forest object. Make a barplot of the predictor importances ranked in decreasing order.

8. With the function `predict`, predict the output of the random forest on the test set and find the test misclassification rate. Comment.

9. Use the package `e1071` and the function `tune.randomForest` to find, with a 10-fold cross validation (this is the default) performed on the training set, which pair of parameters is the best among `ntree=c(500, 1000, 1500, 2000)` and `mtry=c(5, 7, 10, 15, 20)`. Set the option `importance` to `TRUE` so that you can later explore the predictor importances for the best model. Use the functions `summary` and `plot` to interpret the results.

```
library(e1071)
```

10. From the previous question, extract the best model (in the object obtained with the function `tune.randomForest`, it is in `$best.model`). Compare its OOB error and its evolution, its most important predictors and its test error with the random forest obtained with default parameters. Comment.

Comparison between random forests and bagging-of-trees

Exercise 2. The aim of this exercise is to compare random forests with bagging-of-trees from an accuracy point of view. The comparison is illustrated on the dataset `Vehicle`

```
data(Vehicle)
```

(from package `mlbench`), whose purpose is to classify vehicles into four types ("bus", "opel", "saab", and "van") based on features charactering the vehicle silhouettes.

1. Using

```
?Vehicle
```

read the description of the dataset. How many possible predictors does the dataset include and what are their types? How many observations are there?

2. Split (randomly) the data set into a training set (with size 400) and a test set (with size 446).

3. Using the function `tune.randomForest`, train the best random forest that can be obtained by considering a combination of `ntree=c(500, 1000, 2000, 3000)` and `mtry=c(4, 5, 6, 8, 10)`. Keep the variable importances (with the option `importance=TRUE`). Use the options `xtest` and `ytest` to obtain the test error directly. Analyze the results of the tuning process.

Warning: When using the options `xtest` and `ytest`, the forest is *not* kept (hence cannot be used later to make predictions with new data). To use it with the method `tune`, you must then set `keep.forest=TRUE` (otherwise the function will return an error when trying to compute the CV error).

4. Analyze the best forest obtained with the tuning process in the previous question in term of OOB error and test error. Plot the evolution of the OOB errors (as function of the number of trees). Make a barplot of the predictor importances ranked in decreasing order.

5. Using the function `bagging` from the package `ipred`, train a bagging-of-trees classifier on the training set. Evaluate the corresponding OOB error and test error. Compare it with those of the random forest selected above.

6. Using the function `rpart` (from the package `rpart`), train an individual tree classifier on the training set. Evaluate the corresponding test error. Compare it with the test errors associated with the random forest and the bagging-of-trees classifier.